Une technique de catégorisation rapide des requêtes HTTP

Jean Etienne Kiba, Olivier Paul[1]
Institut National des Télécommunications, Département LOR, 9 rue Charles Fourier, 91000 Evry.
E-Mail , Jean-etienne.Kiba@int-evry.fr, : Olivier.Paul@int-evry.fr

Dans ce document nous étudions une technique permettant d'améliorer la performance des outils d'analyse de trafic dans le cas du trafic HTTP. Pour cela nous proposons d'utiliser des paramètres de catégorisation provenant de mesures réalisées dans le réseau. Nous montrons au travers d'un exemple que ces paramètres peuvent, dans une certaine mesure, se substituer aux techniques actuelles utilisant des algorithmes de reconnaissance de motif.

Keywords: Analyse de trafic, détection d'intrusion, pare-feu.

1. Introduction

Ces dernières années ont vu la délégation croissante vers des partenaires externes de fonctions de sécurité qui étaient auparavant implantées dans les entreprises. L'augmentation continue du débit supporté par les liens utilisés dans les réseaux et l'apparition de nouvelles formes d'attaques ont conduit les opérateurs à prendre un rôle croissant dans ces architectures de délégation. Ainsi plusieurs opérateurs fournissent ou s'apprêtent à fournir certaines formes de services de détection et de prévention d'attaques à leurs utilisateurs [8,9]. Fournir de tels services se heurte cependant à plusieurs challenges techniques:

- Les équipements utilisés dans les réseaux actuels possèdent généralement des capacités d'analyse peu développées. Les fonctions les plus fréquemment rencontrées sont des fonctions dérivées des opérations de capture, d'agrégation, de filtrage, d'échantillonnage et de comptage de paquets. De telles opérations sont par exemple utilisées dans un outil tels que Netflow, implanté dans la plupart des routeurs.
- Les nœuds internes des opérateurs transportent généralement des quantités très importantes de trafic. Ainsi le temps qui peut être passé par un nœud afin d'analyser la requête d'un utilisateur est généralement très faible, de l'ordre de quelques centaines de nano secondes. Ainsi les algorithmes utilisés dans les systèmes extrémités ne sont pas utilisables directement. Par exemple l'algorithme de recherche de motif utilisé par l'outil de détection d'intrusion *snort* (Boyer-Moore) a une complexité temporelle en O(n.m) ou n est la taille de la chaîne à analyser et m la taille du motif recherché [1]. De tels algorithmes sont incapables, dans des environnements à très haut débit, de traiter des chaînes d'une longueur supérieure à quelques octets lorsque plusieurs motifs indépendants sont recherchés.
- Les opérateurs sont généralement limités dans l'introduction de nouveaux mécanismes dans leur réseau par deux facteurs, l'un étant la fiabilité de celui-ci, l'autre étant le coût de gestion engendré par ces nouveaux mécanismes. De ce fait,

[1] Ce travail a été financé en partie par le projet GET DDOS et par le projet IST DIADEM.

toute nouvelle technique devrait tirer parti des techniques de mesure et de contrôle existantes afin de limiter la perturbation apportée au réseau.

Dans cet article, nous nous intéressons à un type particulier de communication et cherchons comment ces contraintes pourraient être satisfaites. Nous choisissons de nous limiter au protocole HTTP. Ce protocole est intéressant car les communications HTTP constituent une grosse partie des communications actuelles. D'après les mesures réalisées par certains opérateurs, le trafic HTTP constitue entre 35 et 50% du trafic Internet [2].

L'objectif de cet article est de montrer comment des communications HTTP peuvent être analysées à l'intérieur d'un réseau en essayant de satisfaire aux trois challenges présentés ci-dessus. Nous montrons quelles sont les informations de mesure sur lesquelles notre technique repose. Nous montrons ensuite comment ces informations peuvent être utilisées pour déduire certains paramètres de la communication HTTP. La section suivante montre l'implantation de notre technique comme une extension du module de filtrage IPfilter dans FreeBSD. Nous présentons ensuite le résultat du test de cette méthode en utilisant les modèles et trafic générés à partir d'un serveur web local.

2. Informations de mesure

Notre objectif est de voir dans quelle mesure, les informations de mesure que l'on peut obtenir au niveau réseau ou transport peuvent être utilisées afin d'analyser les communications d'un point de vue applicatif. Dans cette section nous présentons quelles informations sont à notre disposition.

2.1. Le protocole HTTP

Les échanges HTTP peuvent être vus à plusieurs niveaux. Au niveau le plus bas, le protocole HTTP est basé sur une suite d'échanges requête - réponse ou chaque requête définit une opération à réaliser sur un objet situé sur le serveur contacté. Nous appelons ce niveau micro-session. Le format des requêtes et réponses est spécifié pour HTTP 1.1 par le RFC 2616 [3]. Les requêtes et réponses sont organisées sous la forme d'éléments d'information appelés entêtes. Il existe plus de 40 types d'entête dans HTTP 1.1, cependant seulement quelques-unes sont utilisées fréquemment. Les types les plus couramment utilisés dans les messages de requête sont la version du protocole, les langages et types de codages supportés, la méthode indiquant l'action à réaliser, un identifiant URI désignant l'objet sur lequel la méthode s'applique, l'identité du serveur sur lequel se trouve l'objet, l'identifiant du logiciel client réalisant la requête, le type de connexion et la date à laquelle la requête est réalisée.

D'une manière analogue, les messages de réponse contiennent généralement des entêtes similaires (version, système de codage, date, identité du serveur) ainsi que des entêtes propres (statut de la requête, longueur des données, informations relatives aux caches).

2.2. Sélection des informations

Dans cette section nous fournissons une première approximation des liens que nous pensons exister entre mesures de niveau réseau/transport et informations propres au protocole HTTP telles que celles présentées dans la section précédente. Il est évident que certaines informations applicatives (serveur utilisé, type de connexion,...) ont un impact sur les informations de niveau transport/réseau, cependant pour la plupart d'entre eux les protocoles intermédiaires peuvent rendre plus difficile la découverte de ce lien. De plus certains paramètres sont plus difficiles à découvrir que d'autres. Le Tableau 1 fournit une première approximation des relations que nous espérons prouver.

Tableau 1. Relations escomptées entre paramètres HTTP et paramètres réseau.

Paramètre	Paramètres de niveau réseau/transport
Méthode	Taille de la requête, Taille de la réponse
URI	Taille de la réponse
Source	Adresse IP Source, Port
Destination	Adresse IP Destination, Port
Heure/Date	Donnée externe: Heure/Date
Compression	Donnée externe: Configuration du serveur
Résultat	Taille de la réponse.

Comme on peut le constater dans le tableau, les tailles sont supposées fournir une source d'information importante pour déduire certains paramètres applicatifs. Plus précisément, nous supposons que la taille des objets et les identifiants d'objets (URI) sont des éléments fortement corrélés et que d'autre part que les tailles mesurées au niveau transport/réseau et les tailles d'objets sont également corrélées. Cette dernière observation est facilement vérifiable dans HTTP 1.0 car une connexion TCP est associée à chaque micro session. Dans HTTP 1.1 cette observation est plus délicate à prouver car cette version du protocole utilise plusieurs améliorations qui peuvent rendre ces relations plus faibles.

2.3. La relation HTTP 1.1/ TCP

HTTP 1.1 fournit la capacité aux clients et serveurs web de multiplexer plusieurs micro sessions sur une seule connexion TCP. L'objectif est de réduire la charge générée par les ouvertures et fermetures de connexion TCP sur les serveurs HTTP. Ces connexions sont appelées connexions persistantes. Parmi les connexions persistantes on peut par ailleurs distinguer les connexions en mode pipeline et les connexions normales. Lors de l'utilisation de connexions en mode pipeline, le client peut envoyer une série de requêtes au serveur sans attendre de réponse. Cette capacité est cependant limitée par la structure des documents HTML puisque les objets importés ne peuvent être demandés qu'une fois le document les important ayant été chargé. De ce fait les micro-sessions peuvent être découvertes au niveau réseau en cherchant:

- Dans le cas des connexions non persistantes, les ouvertures et fermetures de connexions TCP. La quantité d'informations transportées entre le serveur et le client peut être déterminée au moyen des numéros de séquence TCP utilisés entre le début et la fin de la connexion.
- Dans le cas de connexions persistantes n'utilisant pas le mode pipeline, les motifs requête - réponse comme indiqué en [5]. Ces motifs peuvent être découverts au niveau transport en considérant l'évolution des numéros de séquence. Comme le numéro de séquence depuis le client n'augmente que lorsque qu'une nouvelle requête est envoyée au serveur, nous pouvons définir comme le début d'une micro session le moment ou est reçu un paquet dont le numéro de séquence est supérieur aux précédents. Un exemple d'échange est présenté en Figure 1. Dans ce diagramme temporel les requêtes et réponses HTTP sont indiquées par des flèches sur les cotés. La quantité d'informations transportées entre le serveur et le client peut être déterminée au moyen des numéros de séquence TCP utilisés entre le début et la fin de chaque micro session.

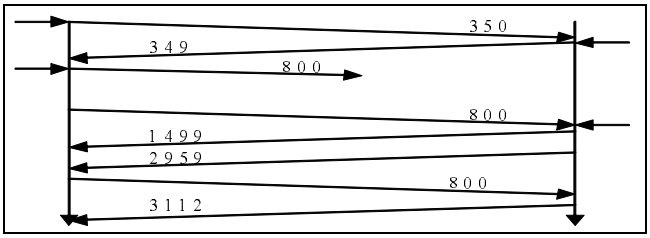

Figure 1. Exemple de micro sessions dans le cas d'une connexion persistantes.

- Dans le cas de connexions persistantes utilisant le mode pipeline les motifs requête - réponse. Dans ce cas seul le premier échange a une valeur intéressante, les suivants pouvant inclure une ou plusieurs micro sessions. La quantité d'information transporté entre le serveur et le client pour le premier objet peut être déterminée au moyen des numéros de séquence utilisés entre le début et la fin de la première micro session.

Comme on peut le constater, les connexions persistantes en mode pipeline rende la relation entre taille mesurée au niveau réseau et taille des objets assez faible. Il est donc intéressant de se demander si, dans la pratique, ce type de connexion est fréquemment utilisé. Afin de répondre à cette question, nous pouvons d'une part étudier les capacités des navigateurs et d'autre part les types de connexions reçues par des serveurs internes à notre organisation.

Tableau 2. Capacités des navigateurs.

Navigateur	Persistance	Pipeline	Part de Marché
MS IE	Oui	Non	77%
Firefox 1.0, Mozilla 1.0	Oui	Inactif par défaut	15%
Netscape	Oui	Inactif par défaut	2%
Opera 7.0	Oui	Oui	2%

Comme indiqué dans le Tableau 2, la plupart des navigateurs ne supportent pas à la fois les modes persistance et pipeline. De plus la plupart des navigateurs supportant ces deux modes n'activent pas le mode pipeline. Ce type de configuration par défaut peut s'expliquer pour deux raisons: D'une part [4] a montré que le mode pipeline bénéficie surtout aux utilisateurs connectés à Internet par un modem téléphonique ce qui comprend une fraction sans cesse décroissante des utilisateurs. D'autre part certains caches web et partageurs de charge ne supportent pas le mode pipeline ce qui conduit à l'inactiver par défaut dans un soucis de compatibilité. Par ailleurs, l'analyse des communications reçues par notre serveur indique que moins de 1% des communications utilisent le mode pipeline.

3. Inférence des données applicatives

Notre supposition dans cette section est que la taille des messages de réponse est dominée par la taille des objets transportés et que la taille de l'entête peut prendre un nombre limité de valeurs. Pour un serveur donné, ces valeurs dépendent de la configuration du serveur.

3.1. Inférence du type de réponse.

Afin de déterminer la relation existant entre taille totale des réponses, taille des entêtes HTTP et la type de réponse sur notre serveur, nous capturons les réponses de notre serveur pendant plusieurs heures. Le résultat est indiqué en Figure 2.

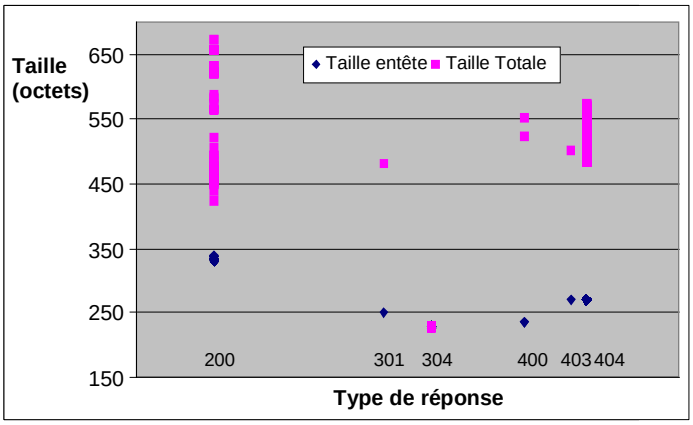

Figure 2. Relations entre tailles et type de réponse (serveur apache 1.3)

Six types de réponse ont été capturés pendant la période de mesure:

- Réponses de type 200 indiquant une réponse correcte.
- Réponses de type 301 indiquant que le document recherché a bougé.
- Réponses de type 304 indiquant que le document recherché n'a pas changé depuis la dernière requête.
- Réponses de type 400 indiquant que la requête n'a pas été comprise par le serveur.
- Réponses de type 403 indiquant une action interdite.
- Réponses de type 404 indiquant que l'objet demandé n'est pas présent sur le serveur.

La Figure 2 montre que les réponses "200" peuvent être distinguées des autres réponses en considérant la taille totale (taille totale supérieure à 550 octets). Certaines réponses "200" ont cependant une taille qui entre en collision avec des réponses d'autres types lorsque la taille de l'objet transporté est trop petit. Cependant une analyse de la taille des objets nous a montré que seulement 1% des objets pouvaient connaître ce problème. Les réponses de type "304" peuvent également être distinguées des autres réponses en considérant la taille totale (taille totale inférieure à 230 octets). Les autres types de réponses ne peuvent pas être distinguées entre elles.

De plus les mesures effectuées montrent que deux types d'en-têtes (et donc de tailles d'en-têtes HTTP) sont utilisées par les clients de notre serveur:

- Les entêtes pour les connexions non persistantes qui incluent les entêtes présentés en section 2.
- Les entêtes pour les connexions persistantes qui contiennent en plus de ces entêtes des informations liées à la persistance de la connexion. La taille de ces entêtes complémentaires est prévisible (environ 57 octets).

Ainsi il est possible de connaître l'influence d'une connexion persistante sur la taille de l'entête HTTP. L'information de persistance peut elle même être obtenue cherchant quel système de délimitation décrit en section 2 s'applique à la connexion considérée. Le Tableau 3 fournit les relations existant entre taille des réponses et le type de réponse en fonction de la taille et de la persistance de la connexion.

Tableau 3. Classification des types de réponse (serveur apache 1.3).

Résultat	**Taille de la réponse**
200 (Non Persistant)	RS >510 ou 200<RS<400
200 (Persistant)	RS >550 ou 260<RS<460
304 (Non Persistant)	190<RS <200

304 (Persistant)	240<RS <250
301, 400, 403, 404 (Non Persistant)	400<RS<510
301, 400, 403, 404 (Persistant)	460<RS<550

3.2. Inférence de la taille des objets.

La Figure 2 montre que les réponses de type "200" peuvent comporter une entête HTTP dont la taille n'est pas fixe. Utiliser une valeur moyenne pour cette taille peut donc conduire à des erreurs dans l'estimation de la taille des objets. Bien que la variance de cette variable ne soit pas très importante nous avons besoin d'une estimation de meilleure qualité. En regardant les entêtes HTTP de plus près il est possible d'ordonner les types d'entêtes en plusieurs classes:

- Certains types d'entêtes ont une taille et une valeur fixe (code de réponse, identifiant du serveur,...).
- Certains types d'entêtes ont des valeurs variables mais une taille fixe (date de dernière mise à jour, Etag, ...).
- Certains types d'entêtes ont une taille qui dépend de l'objet (type de contenu, taille de l'objet).
- Enfin quelques types d'entêtes ont une taille qui peut être prévue par l'observation des communications au niveau réseau (Keepalive).

Ainsi pour un objet et une connexion donnés, il est possible de déterminer la taille de l'entête HTTP relative à l'objet transporté. Ceci signifie qu'il est possible en gardant la correspondance [taille totale mesurée, taille de l'objet] de retrouver les tailles d'objets possibles.

La plupart des serveurs HTTP supportent par ailleurs la compression des objets avant leur transmission sur le réseau. Cette opération peut affaiblir la relation entre la taille totale et la taille réelle de l'objet. Cependant il est toujours possible de savoir si un serveur utilise la compression par l'analyse de ses options d'exécution. Par ailleurs la plupart des serveurs permettent de stocker la taille des objets compressés dans les fichiers de logs en complément de leur taille avant compression. Une analyse de ces logs peut donc permettre de construire une correspondance entre tailles compressées et objets qui nous ramène au cas précédent.

4. Inférence des URIs

Notre supposition est que pour les opérations utilisant une méthode "GET", les paramètres mesurés aux niveaux réseau/transport permettent d'inférer les identifiants URI. Les liens entre ces paramètres et les URIs est évident dans certains cas et moins évidents dans d'autres:

- Chaque objet possède une taille unique. Connaître une URI permet donc d'expliquer la mesure d'une taille et réciproquement.
- Les utilisateurs provenant d'endroits différents ont des centres d'intérêts différents. Par exemple les étudiants présents localement sont généralement intéressés par les informations liées aux horaires et au contenu des cours. Les utilisateurs se connectant depuis l'extérieur sont souvent plus intéressés par les contenus liés aux activités de recherche. Les utilisateurs de pays étrangers sont plus intéressés par les objets en anglais alors que les utilisateurs francophones sont plus intéressés par les pages en français. L'adresse IP du client peut donc expliquer dans une certaine mesure l'URI.
- Pour des raisons similaires la date et l'heure peuvent également avoir une certaine influence.

4.1. Source d'information pour la construction d'un modèle

Afin de modéliser les relations pouvant exister entre paramètres de mesure, nous utilisons les fichiers de logs disponibles sur notre serveur. Ces logs sont généralement constitués d'un ensemble d'entrées, chacune décrivant une opération réalisée sur le serveur. La nature de ces entrées dépend de la configuration du serveur. La ligne suivante est un exemple d'entrée trouvée sur notre serveur :

```
15.17.18.20  -  -  [07/Jan/2004:16:04:14]  "GET
/~michel/TPLDAP.html HTTP/1.0" 200 23154
```

Dans cette entrée 15.17.18.20 représente l'adresse du client ayant effectué la requête. Les deux tirets représentent l'identité de l'utilisateur lorsque celle-ci est connue. Le champ suivant fournit la date et l'heure à laquelle l'opération a été réalisée. Nous trouvons ensuite la requête reçue par le serveur. Celle-ci est une opération de type GET sur l'objet /~michel/TPLDAP.html en utilisant le protocole HTTP 1.0. Les deux champs qui suivent indiquent que l'opération a été réalisée de manière correcte (code 200) et que la taille de l'objet transmis est de 23154 octets.

4.2. Source d'information pour la construction d'un modèle

Le modèle que nous avons choisi afin de représenter les relations entre les paramètres précédemment mentionnés est un réseau bayésien. Les réseaux bayésiens sont des modèles graphiques qui permettent de représenter les relations causales entres des variables et de réaliser des opérations d'inférence. Un réseau bayésien est généralement défini par un triplet:

- Un graphe orienté G, $G=(V,E)$, où V est un ensemble de nœuds et E un ensemble d'arcs.
- Un ensemble de probabilité fini (Ω, Z, P).
- Un ensemble de variables définies sur (Ω, Z, P) tels que:

$$(1)\quad P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i | C(V_i))$$

où $C(V_i)$ est l'ensemble de causes de V_i dans G.

L'inférence dans un réseau de causalité consiste à propager une ou plusieurs informations certaines au sein d'un réseau, pour en déduire comment les croyances concernant les autres nœuds sont modifiées. En fait dans le réseau Bayésien la propagation de l'information est due de façon naturelle à la relation de causalité entre les nœuds. Si le nœud X_i est situé en aval du nœud X_j, mais n'est pas le descendant direct de $X_j, (j < i-1)$, on peut écrire:

$$(2)\quad P\langle X_i \,|\, X_j \rangle = \sum_{X_{i-1}} P\langle X_i \,|\, X_{i-1} \rangle \cdot P\langle X_{i-1} \,|\, X_j \rangle.$$

Si le nœud X_i est un descendant du nœud X_j, l'opération d'inférence est terminée, sinon il suffit de décomposer $P\langle X_{i-1} \,|\, X_j \rangle$ de la même façon, jusqu'à arriver au descendant direct de X_j.

Dans le cas où le nœud X_i est situé en amont du nœud X_j il faut d'abord utiliser la propagation avant à partir du début de la chaîne, pour connaître pour chaque nœud sa

probabilité inconditionnelle $P(X_k)$ pour $(1 \le k \le j)$. Pour cela on peut utiliser la propriété d'inversion de la probabilité conditionnelle :

$$(3)\quad P\langle X_i \,|\, X_{i+1}\rangle = \frac{P\langle X_{i+1} \,|\, X_i\rangle \cdot P(X_i)}{P(X_{i+1})}$$

Ainsi si X_i est l'ascendant direct de X_j, alors l'opération est terminée, sinon, il suffit de continuer de proche en proche.

4.3. Sélection des variables

Afin d'obtenir un réseau bayésien à la fois efficace et compact, il est important de diminuer la taille du modèle. Pour cela il est possible d'une part d'agir sur le nombre de variables prises en compte et d'autre part sur le nombre de valeurs pouvant être prise par ces variables. Pour cette dernière opération nous réalisons une agrégation des valeurs de manière arbitraire:

- Les adresses IP sont agrégées sous forme de codes pays. Cette opération nous permet de passer dans le cas d'IPv4 de 4 Milliards à 225 identifiants.
- Les secondes, minutes et heures sont agrégées sous la forme d'une seule variable donnant l'heure. Cette opération nous permet de passer de 86400 valeurs potentielles à 24.
- Le jour de la semaine, mois et année sont agrégés sous la forme d'une seule variable donnant le jour. Cette opération permet de diminuer le nombre potentiel de valeurs d'un million à 7.

Comme le coût de l'inférence dans un réseau bayésien augmente de manière exponentielle avec le nombre de variables, il est important de limiter le nombre de variables prises en compte. Pour cela nous évaluons la capacité de chaque variable (taille, code pays, heure, jour) à expliquer la variable URI. Pour cela nous comparons pour chaque variable X P(URI∩X) avec P(URI).P(X) en calculant:

$$(4)\quad I(URI, X) = \frac{\sum_{i=1}^{N} \left([P(URI \cap X_i) - P(URI)P(X_i)]\right)}{N}$$

Le résultat montre que les variables taille et code pays sont les variables les plus efficaces (I=0.7, I=0.3 respectivement) alors que les autres variables sont moins intéressantes (I=0.1).

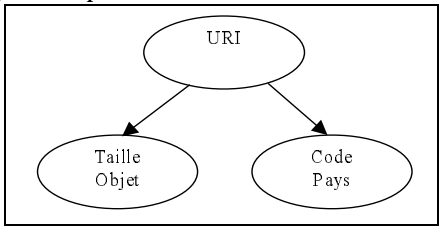

Figure 3. Réseau bayésien sélectionné.

Ceci nous conduit à construire à réseau bayésien très simple comme indiqué en Figure 3. La simplicité de ce réseau permet de réaliser des opérations d'inférence sur l'URI en un temps constant.

5. Implantation et Tests

Une implantation du processus décrit précédemment a été réalisée. La base de notre architecture d'analyse est le module de filtrage IPfilter de FreeBSD. Ce module a été étendu afin de permettre la capture de micro-sessions telles que définies en section 2. Une fois ces micro-sessions reconnues, les données les caractérisant (adresses, ports source et destination, temps de capture des premier et dernier paquets, taille des données transportées dans les deux sens, ...) sont passées au travers de l'interface *syslog* à un analyseur dans l'espace applicatif. Celui-ci détermine tout d'abord si la connexion est persistante. Il détermine ensuite le type de réponse puis, lorsque le type de réponse est estimé correct ("200") effectue les agrégations de valeurs nécessaires à l'adaptation au réseau bayésien. Enfin le réseau bayésien est utilisé afin de déterminer l'URI la plus probable.

Le modèle utilisé pour construire notre réseau bayésien est créé à partir du fichier de logs présent sur un serveur départemental. Celui-ci est constitué d'un serveur apache 1.3 sur une station SUN ultra 10. Il contient environ 15k objets et reçois quelques 10k requêtes par jour. Afin d'obtenir une quantité de données suffisante pour la constitution du réseau bayésien, nous utilisons un fichier de logs correspondant à environ un mois de trafic.

La validation de l'analyseur a été réalisée en effectuant sur le serveur une série de requêtes comparables à celles contenues dans un fichier de logs de 309389 entrées. Comme il ne nous est pas possible de générer des requêtes possédant les adresses IP originales nous avons agit en deux temps. Dans un premier temps, nous avons généré pour chaque objet présent sur le serveur des séries de requêtes en utilisant plusieurs types de navigateurs (MS IE, Firefox, Opera, wget). Nous avons à partir des micro-sessions générées par ces requêtes extrait les tailles mesurées et utilisé ces tailles pour générer des micro-sessions complétées par les adresses IP et informations horaires présentes dans les requêtes des logs. Ces micro-sessions sont considérées comme équivalentes à celles qui auraient été générées par les requêtes présentes dans les logs originaux. Ces micro-sessions ont par la suite été utilisées par le logiciel d'analyse. Les résultats sont indiqués dans le Tableau 1.

Tableau 4. Résultats du logiciel d'analyse.

Paramètre estimé	Réponses considérées	Réponses correctes	Pourcentage correct
Type de réponse	300986	287606	96
URI	154289	135212	87

En terme de performance une implémentation sans optimisation particulière a été testée sur une pentium Xeon 2.4Ghz avec 512Ko de cache de niveau 2 et 1024Mo de RAM. Les tests ont portés sur les performances du réseau bayésien. En effet la partie inférence du type de réponse est extrêmement simple et n'engendre pas un surcoût important. Plusieurs aspects (temps nécessaire à la construction du modèle, taille du modèle, temps moyen nécessaire pour inférer l'URI étant donné la taille mesurée et l'adresse IP du client) ont été étudiés pendant les tests.

Pour cela nous avons de nouveau utilisé le fichier de logs et extrait les valeurs des requêtes ayant un code de retour "200". Nous avons ensuite utilisé ces valeurs pour simuler des requêtes et mesuré le temps de calcul nécessaire pour l'ensemble des requêtes. Une série de 50 tests a par la suite été réalisée et les résultats synthétisés. En moyenne le temps de construction du modèle était de 196 secondes, la taille de celui-ci de 2.5Mo et le temps moyen par requête de 970ns. De telles performances devraient être suffisantes pour analyser des liens à des débits de l'ordre de 20Gb/s en supposant des caractéristiques de trafic typiques de celles rencontrées sur Internet.

6. Discussion et conclusion

Dans cet article nous avons présenté une méthode permettant de réaliser l'analyse de certains paramètres des communications HTTP au travers de mesures réseau. Nous avons par ailleurs montré au travers d'un exemple que cette technique semblait fournir des résultats assez précis dans la pratique ainsi que des performances intéressantes. A notre connaissance il n'est pas aujourd'hui possible de faire des analyses à de tels débits au moyen d'outils utilisant des algorithmes de recherche de motifs [6].

Une question est cependant de savoir dans quelle mesure cette technique peut s'appliquer à des sites ou le nombre d'objets est beaucoup plus élevé. Notre sentiment sur ce point est que la précision de la technique dépend essentiellement du nombre moyen d'objets par taille. [7] indique que la distribution cumulative des tailles des réponses est quasi logarithmique sur 5 ordres de magnitude. De ce fait les collisions entre objets de taille faibles devraient être relativement fréquentes engendrant un faible taux de reconnaissance pour ces objets. Cependant ce problème doit être relativisé. Si on considère en [7] les réponses de taille supérieure à 500 octets (réponses transportant des objets) on obtient un proportion de 0.16% des tailles entre 500 et 1000 octets. Même dans un site contenant 10^5 objets, le nombre moyen d'objets par taille pour cette classe reste de l'ordre de 30. Ainsi si on considère notre technique comme un complément des techniques de recherche de motif, celle-ci peut être utile afin d'effectuer un pré-classement limitant par la suite le nombre de motifs à rechercher et cela pour un coût modeste.

Enfin nous avons utilisé dans cette article une partie limitée des informations permettant de réaliser une inférence. L'utilisation de modèles basés les liens entre objets présents sur un site devrait permettre d'améliorer les résultats obtenus de manière significative.

7. References

[1] *Fast Content-Based Packet Handling for Intrusion Detection*, Mike Fisk, George Varghese, *UCSD Technical Report CS2001-0670, Mai 2001.*

[2] *Sprint IP Monitoring project, available at: http://ipmon.sprint.com/, 2004.*

[3] *HTTP 1.1, RFC 2616. R. Fielding and al. Internet Engineering Task Force, Juin 1999.*

[4] *Network Performance Effects of HTTP/1.1 CSS1, and PNG. 1. H. Nielsen et al.. In Proceedings SIGCOMM 1997, Septembre 1997.*

[5] *PRO-COW: Protocol Compliance on the Web, A Longitudinal Study 5. Balachander Krishnamurthy, Martin Arlitt, , USITS '01, Mars 2001.*

[6] *Hardware-Based Pattern Matching Engines for Network Intrusion Detection Systems*, Sherif Yusuf, *ICL Technical Report, Mai 2004.*

[7] *Tracking the Evolution of Web Traffic: 1995-2003,* Félix Hernández-Campos Kevin Jeffay F. Donelson Smith, *In proceedings of MASCOTS 03, Octobre 2003.*

[8] *DDoS Mitigation via Regional Cleaning Centers, Sharad Agarwaly, Travis Dawson, Christos Tryfonasy, SPRINT ATL RESEARCH REPORT RR04-ATL-013177 - Janvier 2004.*

[9] *Real-Time Mitigation Of Denial Of Service Attacks Now Available With AT&T Internet Protect. ATT news release, Juin 2004. Disponible sur http://www.att.com/news/2004/06/01-13096.*